SCHEDULING ALGORITHMS FOR MESSAGE TRANSMISSION OVER A SATELLITE BROADCAST SYSTEM

Eytan Modiano

MIT Lincoln Laboratory
Advanced Networks Group
Lexington, MA 02173
modiano@ll.mit.edu

Abstract

This paper addresses scheduling algorithms for message transmissions over a satellite broadcast system. The system is expected to deliver messages of widely varied length. Our objective is to find a scheduling algorithm that exhibits good delay performance for messages of all sizes. we show that classical scheduling algorithms such as First-Come-First-Serve and Round-Robin perform poorly in this environment. We study two alternative schemes. The first, gives preemptive priority to the message with the Shortest Remaining Processing Time (SRPT). This scheme is known to minimize overall average message delays, but results in disproportionately large delays for long messages. The second scheme, serves messages based on a dynamic priority function, where a message's priority varies based on how long the message has been in the system as well as its length. This scheme results in somewhat larger overall average delays, but it is more fair to long messages.

I. Introduction

This paper addresses scheduling algorithms for message transmissions over a satellite broadcast system. This problem is motivated by the need to design a message transfer protocol for the Battlefield Awareness Data Dissemination (BADD) system. BADD is an information dissemination system that is being designed to efficiently transfer battlefield information over the "one-way" Global Broadcast Service (GBS). A view of the BADD-GBS system is shown in figure 1. In this system, messages are sent over the Defense Information Systems Network (DISN) to the BADD/GBS transmit site, where they are scheduled for transmission. Once received at the BADD/GBS receive site, they are disseminated to their respective destination. The scheduling algorithms considered here focus exclusively on the transmission of messages over the satellite which takes place at the BADD/GBS transmit site.

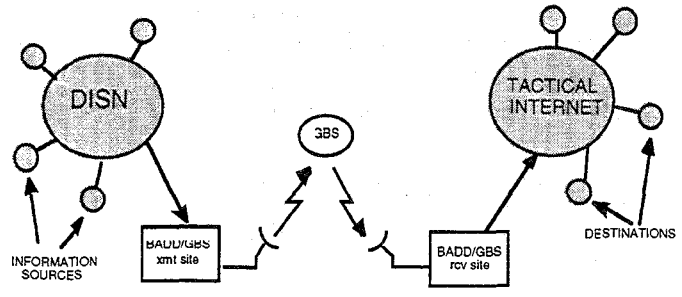

Figure 1. The BADD-GBS system.

Similar scheduling problems have been studied extensively in the context of processor sharing. Much of the theory behind scheduling algorithms has been developed in that context and applies well to the problem of scheduling of message transmissions. Our focus, here, however, is on finding scheduling algorithms that are suited to the BADD/GBS system. Therefore, in this paper we do not attempt to derive optimality scheduling results, but rather to describe and analyze, often through simulation, practical scheduling schemes that are suited to the BADD/GBS system.

The BADD system is expected to deliver messages of widely varied length. Messages vary from the very short (e.g., 100 bytes) to extremely long image files that exceed several Gbits in length. It is this wide variability in message lengths that makes message scheduling both very important as well as a challenge. To illustrate the significance of the problem, consider what happens when a First-Come-First-Serve (FCFS) service discipline is employed and a short packet arrives immediately after a very long 1 Gbyte file. When operating over a 10 Mbps channel, the short message (which may represent a short status message or a protocol control packet) will have to wait almost 15 minutes before it is delivered, at which point it may be totally outdated. The situation depicted above may be remedied by a simple priority scheduling where these very short messages receive preemptive priority over the long messages. However, while such a priority mechanism may work well with just two file sizes of extreme values it may not be easy to extend to more general

0-7803-4249-6/97/$10.00 © 1997 IEEE

cases. For example, how would one handle a 100 Mbit file when it arrives after a 1 Gbyte file; should it have to wait the 15 minutes? If not, would it be fair to interrupt the long file with for almost two minutes in order to transmit the shorter file?

Scheduling algorithms can be measured against a number of performance criteria. The most obvious is average delay; both overall average delay as well as average delay offered to different traffic classes. Another important performance measure is "fairness". As depicted in the previous paragraph a FCFS scheduling policy would be unfair to short files. While there is no universal definition for fairness, a scheduling algorithm can be considered fair if the service time received by a file is proportional to the length of the file. Yet another, less obvious, performance measure is the variability in delay between messages of similar size. In this paper we describe a number of scheduling algorithms and compare their performance using the measures described above. A more comprehensive look at these scheduling algorithms is presented in [1].

In addition, there are other considerations for a scheduling algorithm. For example, certain messages may convey timely information, such as protocol control information, that must be delivered within certain time limits. In this note we do not address this problem explicitly, but only point out that such messages may be best served by giving them higher priority over other messages. Another consideration for messages transmitted over military communication resources is the military precedence class. However, again, we do not address this issue explicitly but rather only consider the problem of scheduling within a precedence class.

Finally, as a matter of practical interest, the scheduling algorithm has to be implemented at the BADD/GBS transmit site. This approach is primarily dictated by the need to be compatible with the current GBS system where messages are first transferred to the GBS server and then are scheduled for transmission. As a first design, we view this scheduling protocol as a higher layer protocol (e.g., application layer), where the application, in this case the BADD message transfer protocol, is responsible for scheduling the message transmissions over a single GBS link. In the future, it will be interesting to consider scheduling algorithms that schedule messages at the network layer, beginning at their source all the way to their destination, rather then only over the GBS link.

II. Scheduling Algorithms

In this section we describe a number of scheduling algorithms. These algorithms can schedule the messages based on message characteristics such as length, the amount of time that the message has been in the system and the amount of service so far received by the message. The majority of algorithms that we consider schedule messages once, when they arrive, based on message characteristics. We call such scheduling algorithms "static" because the priority of a message does not change in time. We also consider scheduling algorithms that alter a message's priority value based on the amount of time that the message has spent in the system, or the amount of service already received by the message. We call these scheduling algorithms "dynamic" for the obvious reason. Finally, scheduling algorithms can be either preemptive where message transmissions can be interrupted by higher priority messages or non-preemptive. We consider both preemptive and non-preemptive algorithms.

A. First-Come-First-Serve

A simple and intuitive scheduling algorithm is a FCFS algorithm. FCFS is a non-preemptive, static algorithm. In FCFS, messages are sorted based on their time of arrival and are served, to completion, in order of arrival. As discussed in the introduction, FCFS has the obvious shortcoming of being unfair to short messages. This will be shown explicitly in the next section. One interesting fact about FCFS scheduling is the simplicity of the analysis. FCFS scheduling algorithm can be modeled using an M/G/1 queueing system, for which closed form results on average delays are readily available in closed form [2,3]. These FCFS results can be used as a basis of comparison for the various algorithms. Eq.(1) gives the average message delay for a system using FCFS scheduling.

$$D_{FCFS} = \frac{\lambda \overline{X^2}}{2(1-\rho)} + \overline{X} \qquad (1)$$

where X is the message service time (e.g., transmission time), l is the message arrival rate and r is the channel utilization. Therefore the above expression gives the average message delay in terms of the first and second moments of the message transmission times. Of course, the message transmission times are simply the message length divided by the transmission rate of the channel.

B. Round-Robin

In a round-robin scheme messages are broken down into small "cells" (e.g., ATM cells) and are served one cell at a time in a round-robin fashion, rotating among the messages. Inherently, round-robin schemes are preemptive, though not based on some pre-defined priority. A round-robin scheme overcomes the shortcoming of FCFS because the short message would not have to wait for the complete transmission of a long message. However, round-robin has other shortcomings in that it results in unnecessarily large delays in situations where there are many messages in the system. More about the performance of round robin will be said in the next section.

Just as with FCFS, average delay results for round-robin scheduling are available in closed form [4]. Eq.(2) gives the average delay for round-robin scheduling. The "fairness" of round-robin scheduling can be immediately seen from the equation where the average delay is clearly proportional to the message length.

$$D_{RR} = \frac{\overline{X}}{1-\rho} \qquad (2)$$

where again p is the channel utilization and X is the message transmission time.

C. Bucket scheduling

Bucket scheduling schemes are a variant of round-robin schemes which attempt to rectify the problem that arises when pure round-robin is used with a large number of files. In bucket scheduling, messages are sorted into buckets based on their size and the buckets are served in a round-robin manner. Once again the messages are divided into cells and the buckets are served a number of cells at a time. The number of cells served from each bucket can vary between buckets. A general bucket scheme is shown in figure 2. Each of the buckets contains messages of a given size or range of sizes and the server serves each bucket in a round robin fashion; serving Di cells from bucket i. Bucket schemes are of particular interest because in the preliminary GBS system a bucket scheme was employed. That scheme used 3 buckets, one for urgent data, one for messages smaller than 50Mbytes and one for messages larger than 50Mbytes. The server dwelled on the urgent bucket for up to 1000 cells and on the short and long buckets for 2 and 1 cells respectively. This in effect gave the urgent data almost preemptive priority over the rest of the data.

The difficulty in designing a bucket scheme is that there are so many possibilities for the number of buckets, the range of packet sizes that are input to the buckets and the dwell times on the buckets. Also, unfortunately, bucket schemes are difficult to analyze in closed form and their analysis requires the use of simulation, further limiting our ability to gain insight into the system. As will be discussed in the next section we were able to obtain some idea about the performance of bucket scheme by considering a limited number of cases for which we could find the ideal bucket scheme through the use of simulation.

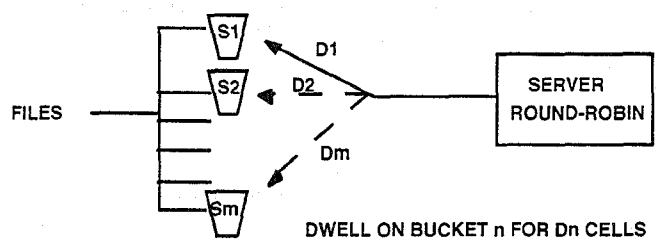

Figure 2. A bucket scheduling scheme with round-robin server.

D. Shorter Messages Preempt (SMP)

SMP is a simple preemptive priority scheme where shorter messages receive preemptive priority over long messages. In this scheme all messages are broken down into cells and every cell is "tagged" with a priority value that is equal to the length of the original message to which the cell belongs. The server then serves the cells, one at a time, giving highest priority to the cell with the smallest tag value. This, in effect, amounts to giving shorter messages preemptive priority over longer ones.

The performance of this scheme can be approximated with that of a pure preemptive priority scheme. A pure preemptive priority scheme is one where the shorter messages would preempt longer ones as soon as they arrive. This scheme is slightly different because the preemption occurs only at the end of transmission of a cell. With cell sizes that are relatively small compared to the message size the analysis of pure preemption provides a good approximation to the cell based scheme. The delay results for pure preemptive priority are given in [2,3].

Suppose that messages arrive in finite number of length's $X_1,...,X_n$ and arrival rates $l_1,...,l_n$. Also assume without loss of generality that $X_1 < X_2 <....< X_n$ then it is clear that packets of length X_i have higher priority than packets of length X_j when $i < j$ and lower priority when $i > j$. The average delay, including service time, for messages of length X_k is given by,

$$D_k = \frac{X_k(1-\rho_1-\cdots\rho_k)+R_k}{(1-\rho_1-\cdots\rho_{k-1})(1-\rho_1-\cdots\rho_k)} \qquad (3)$$

where $\rho_i = \lambda_i X_i$ and $R_k = \sum \lambda_i X_i^2/2$. In section III we will apply this result to the analysis of the SMP scheme.

E. Shortest remaining processing time (SRPT)

A slight variation on the SMP scheme which is known to result in better performance, is a scheduling algorithm that serves the message with the shortest remaining processing (in this case transmission) time. With this scheme, if a

message is being served and a shorter message arrives, the longer message will continue to receive service if the amount of time remaining to completion is smaller than the shorter message. It can be shown that this scheme minimizes overall average delay for all scheduling schemes [4]. However, since this scheme is very similar to SMP and is known to result in smaller delays, we can use the results of SMP scheduling to provide an upper-bound on the delay for the SRPT scheme.

F. Dynamic scheduling

Dynamic scheduling schemes alter the message's priority value based on how long it has been in the system and how much service it has received. The SRPT scheme described above is a simple example of a dynamic scheme because a message priority increases as it is being served. The dynamic scheme that we consider here is slightly more elaborate. A "priority" function is used to assign messages their priority values. At each iteration of the algorithm, the scheduler serves one cell from the highest priority message and recalculates the priority of all of the messages.

The difference between dynamic scheduling schemes is in the structure of the priority function used. We examined two priority function, both of which take into account the length of the message and the amount of time that the message has been in the system. The two priority functions $q_1(t)$ and $q_2(t)$ are given by eq.(4) and eq.(5) respectively.

$$q_1(t) = \frac{t-\tau}{l} \qquad (4)$$

$$q_2(t) = (t-\tau) - l - s \qquad (5)$$

where t is the current time, t is the message arrival time, l is the message length and s is the amount of service already received (all measured in cells). As can be seen eq.(4) and eq.(5) both functions give higher priority to short messages and increase the priority for messages that have been in the system for a long time. In addition, the second function also reduces the priority of messages that already have received some service. A more complete description of these dynamic scheduling schemes is given in [1].

Unfortunately, the performance of these schemes cannot be easily computed analytically; as a result we resorted to simulation to analyze the behavior of these schemes. In section III we summarize these performance results and compare them to the performance of the static schemes.

III. Performance analysis

In this section we study the behavior of the different scheduling schemes in terms of average delays. For three of the five schemes we were able to obtain closed form expressions for delay. However, the bucket schemes and the dynamic schemes were not analytically tractable. To analyze those we resulted to simulation. The simulation was written both in plain C code as well as using the Opnet and BONES network simulation tools. The simulation was performed in these different environments primarily to compare the environments and evaluate available commercial simulation products. Here we will just present the simulation results.

All of the scheduling algorithms described in section II, with the exception of the bucket scheme, can be analyzed either in closed form expression (FCFS, SMP, RR) or through simulation (Dynamic scheme). For the bucket scheme the analysis is complicated by the choice of parameters, such as the number of buckets and the dwell on each bucket. In this simplified situation, with only two traffic streams, an obvious choice is to use two buckets; one for large messages and one for small messages.

Unfortunately, there is no simple way to determine the optimal dwell setting on each bucket. Therefore we simulated the bucket scheme with a number of different dwell settings on the buckets in an effort to find the dwell settings that achieves the best performance. What we generally found was that the best dwell setting for a scheme with two buckets is to dwell for much longer on the short message bucket than on the long message bucket. A dwell setting of 10:1 or 100:1 almost always yielded the best results. One interesting observation is that dwelling on the long message bucket for more than one cell almost always yielded longer average delays. This is because it resulted in larger delays for the short messages. In general, a scheduling scheme that serves short messages first results in better average delay. Essentially this is what we found here as well. The results presented for the bucket scheme, here, only show the performance with the best dwell setting that we could find.

In Section II we presented 2 possible priority functions for use with the dynamic scheme. Our simulation results show that the priority function of eq.(4) was generally superior to that of eq.(5). Therefore here we only show the performance of the dynamic scheme using the priority function of eq.(5). The results of the simulation study of the dynamic scheme with the two different priority functions are presented in [1]. An interesting extension to this work would be to study the performance of the dynamic scheme with various other priority functions.

The primary objective of the analysis is to examine the performance of the scheduling algorithms when the mix of incoming traffic consists of messages of widely varied sizes. To obtain insight into this behavior we start by considering a situation where only two message sizes exist. Small messages (consisting of one cell) and large messages (consisting of 100 cells). We then proceeded to evaluate the

performance of the schemes under three channel loading conditions; medium loading of 50%, high loading of 90% and very high loading of 99%. Results for light loading conditions are not presented because under light loading conditions the delay at the server is minimal and therefore the performance of the scheduling algorithm is of little consequence. For each of the loading situation we considered three traffic mixes. An even mixing where half the load comes from small messages and half from large messages; a mixing where most of the load is from small and finally a mix where most of the load is from large messages. In addition we considered a situation with an 80% load and files of 4 different sizes. All together this amounts to 10 different loading combinations. The results obtained for these different loading situations and their interpretations are shown in figures 3 to 12.

Figures 3-5 compare the performance of the algorithms under medium loading conditions (50% load). We see from the figures that the SMP, bucket and dynamic schemes all perform essentially the same. The RR and FCFS schemes perform substantially worse. This is because in the latter two schemes short messages, which dominate the load, sometimes have to wait for long messages. This results in longer average delays. Notice, however, that FCFS results in substantially better delay results for long packets. Also, results for FCFS are omitted in certain situations because they are far worse than the other schemes and the large delay numbers are far off the scale and cannot be displayed on the graph.

Figures 6-8 compare the performance of the algorithms under heavy loading conditions (90% load). The results are very similar to those obtained under medium loading. The SMP, bucket and dynamic schemes generally perform better than the FCFS and RR schemes. The dynamic scheme performed slightly worse for small packets when there is a heavy load of large packets. This is because when a large packet is in the buffer for a long time it may receive priority over small packets, resulting in large delays for small packets. FCFS performs best for long packets, but has an overall high average delay. This is because it offers essentially the same performance to small and large messages and while this delay is good for the large messages it is very high for short messages. Again, in many instances we omit the FCFS results because the large delay offered to short packets is way off the scale and cannot be displayed on the graph.

Figures 9-11 compare the algorithms under very heavy loading (99% load). Under extremely heavy loading conditions overall delays are relatively large. However, the results obtained are still interesting because they give us insight into the behavior of the system at a time when the system is generally busy. Under these circumstances the difference between scheduling schemes is more noticeable. What we see from the figures is that RR and FCFS yield an overall poor performance. Again the SMP, bucket and dynamic schemes perform similarly and the dynamic scheme, while overall is sub-optimal, offers a compromise between a SMP scheme which is biased toward small messages and a FCFS which is biased toward large messages. Finally figure 12 compares the performance of the algorithm with four different message sizes (1,10,100, and 1000 cell messages) and an 80% load evenly divided between the messages. For all messages except the largest, the SMP scheme performed best. The bucket scheme was almost as good, but we note that finding an optimal dwell setting was non trivial. In this case a dwell setting of 100 on the small bucket, 4 on the bucket of 10 cell messages, 2 on the bucket of 100 cell messages and a single dwell on the 1000 cell messages bucket. This setting was determined by simulating a large number of alternative settings. Another interesting observation is that the dynamic scheme, although not optimal in overall delay, offers fairness in the sense that all messages receive a similar transfer rate. This fairness is also present in the RR scheme, but there the overall performance is very poor.

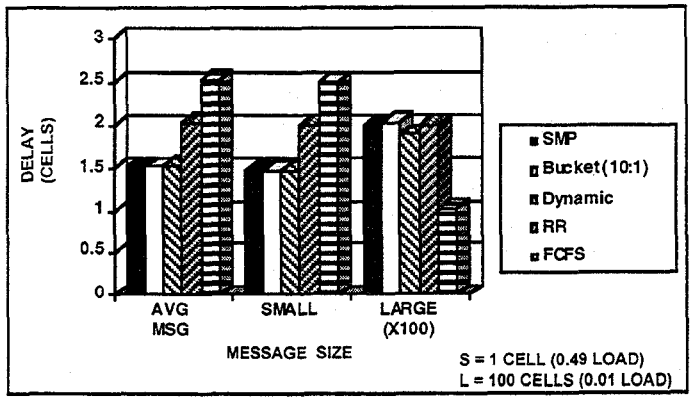

Figure 3. Delay vs. packet size for a load of 50%, dominated by small messages. The dwell setting for the bucket scheme is shown in the legend as (10:1) representing a dwell of 10 cells on the small message bucket and 1 cell on the large message bucket.

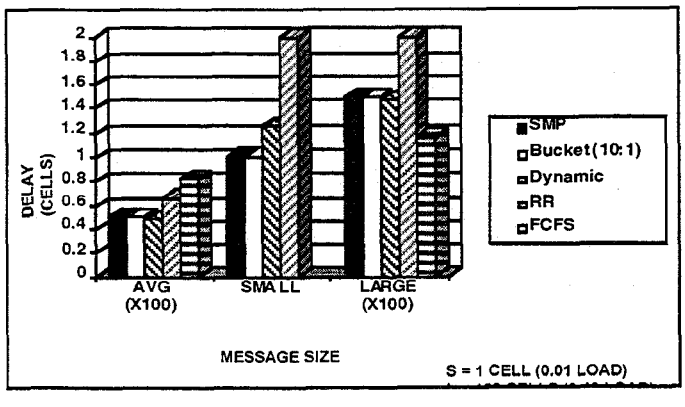

Figure 4. Delay vs. packet size for a load of 50%, dominated by large messages.

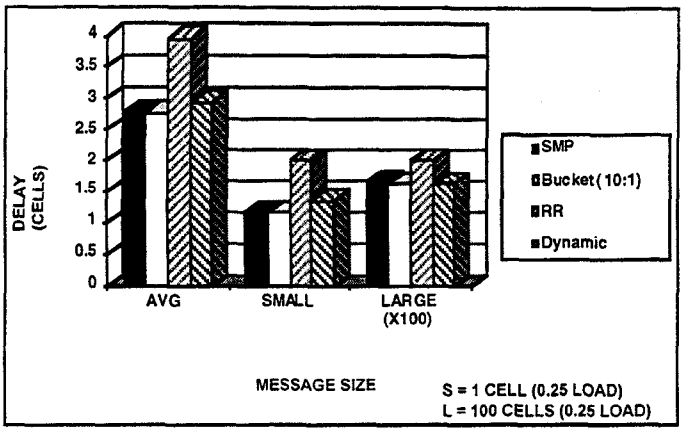

Figure 5. Delay vs. packet size for a load of 50%, evenly divided between large and small messages.

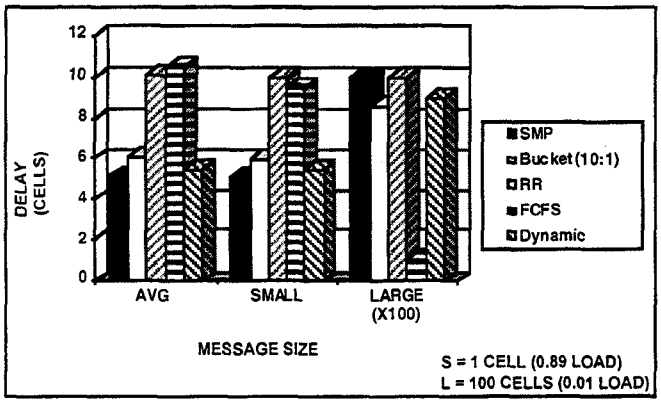

Figure 6. Delay vs. packet size for a load of 90%, dominated by small messages.

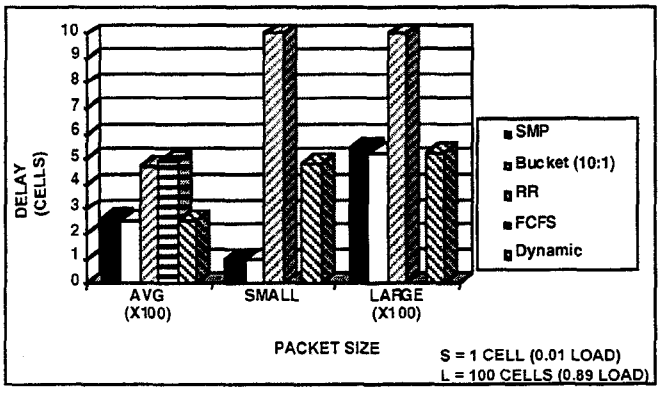

Figure 7. Delay vs. packet size for a load of 90%, dominated by large messages.

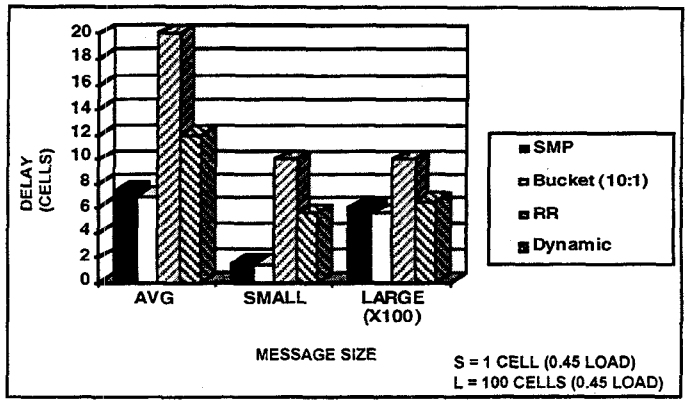

Figure 8. Delay vs. packet size for a load of 90%, evenly divided between small and large messages.

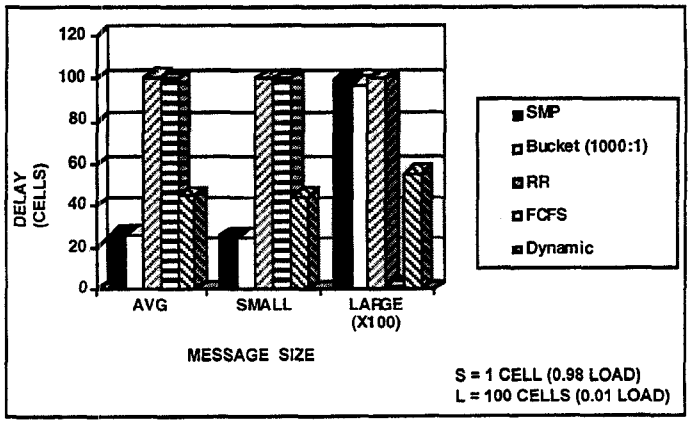

Figure 9. Delay vs. packet size for a load of 99%, dominated by small messages.

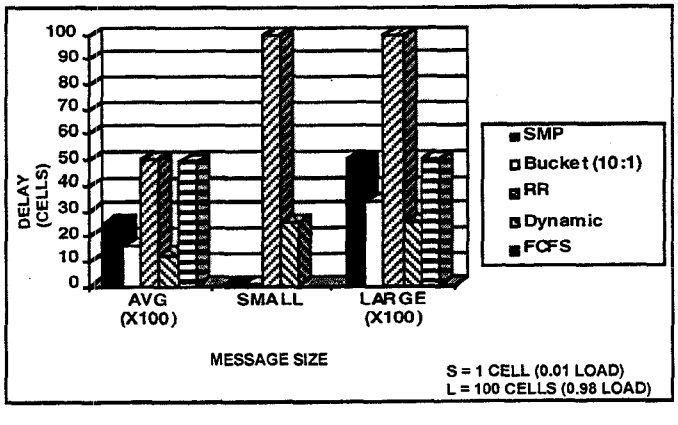

Figure 10. Delay vs. packet size for a load of 99%, dominated by large messages.

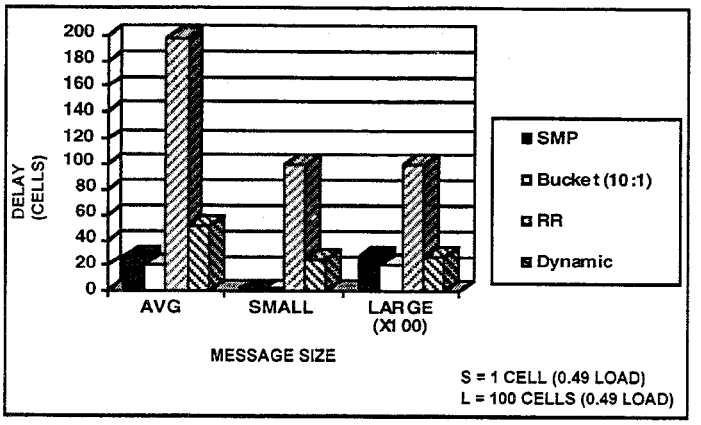

Figure 11. Delay vs. packet size for a load of 99%, evenly divided between small and large messages.

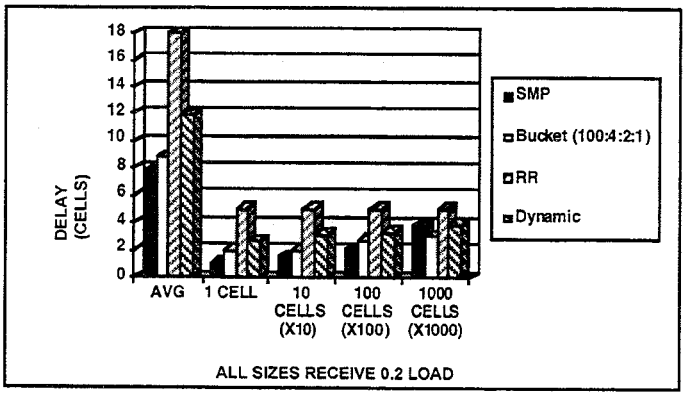

Figure 12. Delay vs. packet size for a load of 80%, evenly divided among four message sizes.

IV. Conclusion

We evaluated a number of scheduling schemes. Of those, Short-Message-Preempt (SMP) generally offered the best average delays. A simple variation of SMP that is known to improve on the performance of SMP is one that always serves the message with the shortest remaining processing time (SRPT). The SRPT scheme is known to minimize overall average delays over all possible scheduling schemes.

The problem with both SMP and SRPT is that they can be unfair to the larger messages. However, none of the other schemes that we analyzed was able to improve performance for long messages without significantly increasing overall average delays. Furthermore, in network traffic, it is likely that short messages contain timely control information or interactive traffic; and therefore, reducing the short message delay can have added benefits in terms of network performance. We therefore believe that the SRPT scheme should be considered for scheduling messages over the BADD/GBS system.

The dynamic priority scheme has the potential of yielding good results for all traffic types. However, as it is defined now, it is clearly sub-optimal. It may be possible to make improvements to the dynamic scheme by altering the priority function. One possible alteration to the priority function is to account for cells already served. Another possibility is to change the slope of the function. Further work in this area would be needed in order to make the dynamic scheme an attractive alternative to the SMP scheme.

Other scheduling schemes that were examined were either inferior in terms of performance or not practical. Both First-Come-First-Serve and Round-Robin scheduling result in very large average delays. The Bucket scheme generally resulted in good delay performance, as long as the optimal number of buckets and dwell settings were used. However, we know of no practical mechanism for determining these values, which makes an efficient implementation of the scheme impractical.

An interested extension of this work to a multi-hop packet switched network is presented in [5] where it is shown that giving priority in a packet switch to packets belonging to shorter messages over those belonging to larger messages can significantly reduce average end-to-end message transmission delays.

References

[1] Eytan Modiano, "Scheduling Algorithms for Message Transmission over the GBS Satellite System," MIT Lincoln Laboratory Technical Report, TR-1035, June, 1997.
[2] Bertsekas and Gallager, Data Networks, prentice Hall, 1992
[3] Kleinrock, Queueing Systems, Vol. 1, John Weily and Sons, 1976.
[4] Kleinrock, Queueing Systems, Vol. 2, John Weily and Sons, 1976.
[5] Eytan Modiano, "Scheduling Packet Transmissions in a Multi-hop Packet Switched Network Based on Message Length," MIT Lincoln Laboratory Technical Report, TR-1036, June, 1997.